SEVEN SECRET TIPS
TO BUILD INTELLIGENT ENTERPRISE ARCHITECTURES

Joseph W. Carl
Riverside Research Institute
Beavercreek, Ohio

John M. Colombi[1]
Air Force Center for Systems Engineering, Air Force Institute of Technology
Wright-Patterson Air Force Base, Ohio

1.0 INTRODUCTION

An article in the Fall 2006 issue of INCOSE *Insight* about the 2007 INCOSE International Symposium [Insight, 2006] cited four behaviors needed for an enterprise to exhibit intelligence: (1) It measures both the value of results to stakeholders and the conformance of results to principles of systems and society; (2) It maintains precise awareness of the enterprise situation with respect to goals; (3) It adapts to and aligns the enterprise with changes in context and capabilities in pursuit of improving goal achievement and sustainable worth; and (4) It ensures enterprise integrity even when all factors are changing unpredictably.

Unfortunately, knowing that these behaviors signify an intelligent enterprise does not tell us how to build intelligent enterprise architectures. But there are seven secret tips gained through hard experience that can substantially contribute to the construction of successful enterprise architectures. Well, they're really not secret, but if they were widely practiced we wouldn't hear Cobb's Paradox [Cobb, 2004]: "We know why projects fail, we know how to prevent their failure ─ so why do they still fail?"

2.0 SEVEN SECRET TIPS

These seven "secrets" are based in part on an earlier paper [Wieser, et al., 2006] that one of us coauthored. That paper cited seven heuristics but did not provide a description of the tools and techniques that are available to implement the heuristics and overcome the difficulties of building intelligent enterprise architectures. We are now going to share the remaining "secrets" with you.

2.1 Ignorance of Domain Causes Architecture Pain

It is widely known that written and spoken language involves certain difficulties whose root causes have been discussed in the systems engineering literature [Carl, 2005]. This makes it clear that different sub-groups use language differently, which significantly complicates achieving mutual understanding. The differences accrue over time. For example, in 1959 differences in language usage were thought to be so significant that humanists and scientists had lost the ability to communicate effectively with each other [Snow 1998]. That argument continues to receive attention in humanities literature.

Thus, when a systems engineer begins to discuss the needs of an enterprise with those who have been members of that enterprise long enough to be considered experts it is not unusual for there to be some deficiency in the level of mutual understanding. The enterprise experts speak with the language they have come to understand and use every day, while the same can be said for the systems engineer. The problem is that these are not the same sub-groups. The simple fact is that the systems engineer may not be expert in the organizational structure, processes, and operations of the enterprise. So what can be done?

"Secret" tip number one says there are three things we can and must do in order to understand the domain that the enterprise represents: conduct site surveys and interviews; organize and conduct user working groups; and involve stakeholders in early and frequent discussions about the evolving architecture and the thinking that lies behind the unfolding plans.

---

[1] The views expressed in this article are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 2007		**00-00-2007 to 00-00-2007**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Seven Secret Tips to Build Intelligent Enterprise Architectures	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Air Force Institute of Technology,Air Force Center for Systems Engineering,2950 Hobson Way,Wright Patterson AFB,OH,45433**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
17th International Symposium of the International Council on Systems Engineering (INCOSE ?07), San Diego, CA, 24-28 Jun 2007.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**12**	

Site Surveys and Interviews

When transforming an enterprise there is a standard three-step process to follow: (1) determine the as-is enterprise; (2) define the to-be enterprise; and (3) write a transition plan for the trip from now to then. It is important to visit the enterprise's operational sites and to interview the enterprise experts in order to comprehensively determine the as-is enterprise and gain insight into the to-be enterprise. One of us once went with a team of engineers to Ascension Island to perform a site survey[2]. A hurricane formed between Ascension Island and home, and aircraft were prohibited from flying in a large area of the Atlantic Ocean. So the survey lasted six days when it was scheduled for two. (The program manager gathered the group together on the fifth day and said he had some good news: we could change our underwear. Harry could change with John; John could change with Al; …) Nevertheless, the team gathered a great deal of very useful data to support the modernization of the Eastern Test Range. If the team had not made the site visit they would not have been informed of the unique operations at Ascension Island, the unique equipment configurations they maintained, and the unique environment in which the enterprise worked. The additional time spent on Ascension contributed to the team's deeper understanding. That is only one example of the importance of performing site visits and interviewing enterprise experts to understand in which environment the new enterprise architecture must operate.

User Working Groups

Experience demonstrates the value of organizing and conducting working meetings attended by users who operate the current enterprise and know what problems the new enterprise architecture should solve. When engineers sit down with users the engineers learn a great deal about the operating environment, about unique aspects of the enterprise, and about the problems that users face and wish to resolve. This enables a more precise understanding of architecture requirements and constraints. It is wise to have user working group meetings on a regular basis until the architecture is fairly mature. This provides an opportunity to involve the users in a cooperative effort that enables users to see and influence the direction the enterprise is heading, and enables the architects to get timely feedback to refine their concepts for the new enterprise architecture.

Stakeholder Involvement

An enterprise stakeholder is anyone who has an interest in and derives personal benefit from the enterprise. This includes the owners or the stock holders, the managers, the workers, the consultants, the material suppliers, the service providers, and in some cases the community. The members of a community are stakeholders when the enterprise provides corollary jobs for people in the community (such as restaurant owners, and clothing cleaners), and when the community's environment is potentially threatened by the enterprise (as in environmental impact assessments). Stakeholders not only benefit from the enterprise, they help shape it. Understanding the constraints that are imposed by the stakeholders is an aspect of understanding enterprise requirements. Stakeholders often should have a way to make themselves heard and a way for them to ascertain how things are going. Stakeholder meetings and briefings, and public information campaigns are just three of the ways to get the word out to stakeholders, and time at the meetings should be reserved for questions and answers. In discussions with stakeholders, it will prove best to have the architects present that have domain knowledge ([Grady, 2006]; [Colombi, et al., 2006]).

Thus our message in this first "secret" tip is that failure to perform the three described activities will lead to at least partial ignorance of the enterprise domain, and this will almost certainly cause the architect pain.

2.2 If You Can't Talk the Talk, Then You Can't Walk the Walk

In light of the sub-group aspect of language usage, we should not be surprised to find that statements of requirements frequently contain several kinds of complications that interfere with mutual understanding. A well known heuristic states, "Don't assume that the original statement of the problem is necessarily the best, or even the right one." In fact, missing requirements, volatile requirements, wrong requirements and poorly articulated requirements are a major cause of project cost and schedule over-runs [Kasser, 2002]. Poor requirements yield poor results. Compared to finding problems with requirements in the requirements phase of a programs life cycle, it costs five times more to find and fix missing or wrong requirements during the design phase; ten times more during the (software) coding phase; twenty times more during unit testing; and 200 times more if they remain until delivery and system operation. Finding and fixing requirements errors consumes 70-85% of project rework costs. The more complex is a project at hand, the less is the chance for on-time at-cost delivery. Of the top ten most-complex projects

[2] Ascension is a British island with an airfield that enables arrival and return home; it is about 300 miles off the coast of Africa.

studied by the Standish Group (as reported in their "CHAOS Report 2004"), eight were over schedule by a factor of 1.6 and over budget by a factor of 1.9; the other two were cancelled and never delivered anything, despite significant outlays of money and the passage of time. Problems with the understanding of requirements and their volatility were root causes of the difficulties in most of the cases studied.

The experience of one of us includes the requirement within a customer's system specification for a replacement system to do "everything the current system can do." But the supplier did not know everything the current system could do, and it turned out neither did the customer. So it took exploration during the project to uncover the real requirements and state them in a testable form. Before the project concluded there were volumes of derived requirements flowing from this one stated requirement, and significant schedule and cost overruns. Here's another example: what does the word *modern* mean? As Catherine Belsey [Belsey, 2002] reminds us, it seems to depend on context, and may in fact mean nothing more than *new*. After all, *modern* history is concerned with events after 1600; *modern* languages are differentiated from classic languages that are thousands of years old; *modern* furniture and *modern* art most likely belong to the 20th century. We see that *modern* does not correspond to a specific timeframe.

So what can we do? We can write down our terms of reference, organize requirements working groups, control requirements changes with discipline, and diligently practice the strong principle of communication.

Terms of Reference

Within the Department of Defense (DoD), a lack of common terminology was highlighted as a problem during Operation Desert Storm. This reminds us that one of the first things to do on any architecture development effort is write down the terms of reference: define the words that have jargon meaning in the work environment, and include a definition of all the acronyms and abbreviations that are common in the enterprise environment. There has been a recent responsive push from DoD leadership to develop more common terms of reference amongst the services to address the DoD terminology problem. Joint Pub 1-02, Department of Defense Dictionary of Military and Associated Terms is an excellent reference that defines common terms of reference across the DoD enterprise community.

The lack of common terminology is ubiquitous. Service-specific languages have traditionally been a problem stemming from each service's unique culture. In DoD special operations, for example, there are many terminology differences between Air Force, Army, and joint vocabulary. The simple concept of Battle Damage Assessment (BDA) is a commonly-used term among Air Force fighter pilots. Although all SOF Missions do not produce BDA reports, all produce after-action reports or Operation Summaries (OPSUMs) that often include BDA information SOF Air Mission not only suffers from service unique vocabularies, they also suffer from theater specific vocabularies. Terms used in one theater of the world do not necessarily match those used in another theater. For example, the definition of "mission intent" was used in certain theater joint guidance, but the Air Force in general used the term "initial assessment" and other services use the term "mission analysis." The architecture can become less relevant to some as ill-considered terms of reference are selected to identify elements within the enterprise architecture.

Another useful way to discover and refine terms of reference is through an enterprise's various *communities of interest* (COI). According to [DoD-CIO, 2005], "COI is a term used to describe any collaborative group of users who must exchange information in pursuit of their shared goals, interests, missions, or business processes, and who therefore must have shared vocabulary for the information they exchange."

Requirements Working Group

A requirements working group has two pivotal roles to play. Its first role is to work hard to capture the right requirements and assure that they are understood by all stakeholders the same way. The working group should include representatives from systems engineers, design engineers, human factors engineers, test engineers, users, acquiring agents – everyone who is directly concerned with paying for, developing, qualifying or eventually using or maintaining the system. Gaining the perspectives of all these representatives is crucial to getting the requirements right as soon as possible at each level that requirements exist (Figure 1, after Figure 6.1, page 122 of [Buede, 2000]). And we saw above in this section that failure to get the requirements right leads to costly and time-consuming problems. But requirements rarely stay constant. So the requirements working group has a follow-on role to play as part of a change management process. Whenever a requirements change is proposed, no matter by who, the proposed change should be considered by the requirements working group to consider the impact of the change, and to make recommendations about the proposed change to an executive-level change management control board.

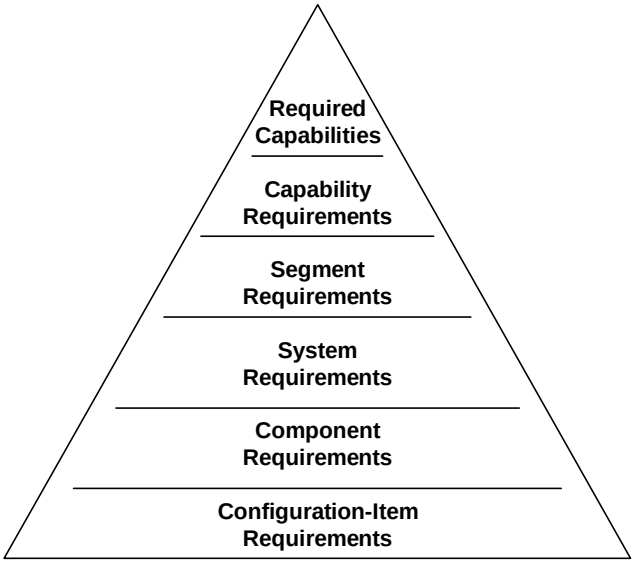

Figure 1: Requirements Exist in Several Categories at Several Levels of Abstraction

Change Management Control Board

Although systems engineering is most accurately about technical integrity, it is often said, "the product of systems engineering is baselines." One of us had a senior systems engineer tell him that systems engineering is only responsible for the technical documents about the system. They indeed are, but the purpose of documented baselines is to achieve technical integrity, and it follows that appropriately identifying, documenting, communicating and managing change is an important aspect of enterprise architecting. This is perhaps most important during concurrent development, and has frequently arisen in various space enterprises. Consider a new constellation of satellites that must be integrated with existing legacy satellites, with the satellite control network, with mission control, and with ongoing developments across various organizations concerned with ground segments of the enterprise architecture (Figure 2). Enterprise change management becomes extremely challenging in this complex enterprise environment.

A change management control board should be invested with executive authority, and should be chaired by a member of the acquisition organization ─ the element of the enterprise that is paying the bills and worrying about the schedule. The system supplier often serves as the secretariat of the board. All deliberative bodies that evaluate proposed changes, such as the requirements working group, provide support, advice, and recommendations to the control board, but only the executive authority invested in the control board can decide when a change can be allowed and when it cannot.

Strong Principle of Human Communication

The strong principle of communication states that any two parties in a conversation must both work as hard as they can to understand what the other is trying to say. But it is straightforward to demonstrate that we do not listen (or read) passively; we actively involve ourselves in the interpretation of what is being said (or written), and anticipate the meaning of what others are trying to tell us. Here is an example that demonstrates readers are actively involved in the synthesis of meaning when presented with a text:

> "*I cdnuolt blveiee taht I wluod aulaclty uesdnatnrd waht I was rdanieg. The phaonmneal pweor of the hmuan mnid! Aoccdrnig to rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoatnt tihng is taht the frist and lsat ltteer be in the rghit pclae. The rset can be a taotl mses and you can sitll raed it wouthit a porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe. Amzanig huh? Yaeh and I awlyas thgohut slpeling was ipmorantt*!"

Even if you don't find this to be a compelling example, remember that a systems engineer should understand that the sub-cultural differences among the participants in a system or product acquisition can lead to different interpretations and meanings associated with words. The systems engineer should not assume that every requirement is understood the same way by all stakeholders, nor that she or he understands every nuance of meaning in the enterprise requirements expressed by another. We all have to learn to suppress our egos and listen more empathetically to what others are trying to tell us.

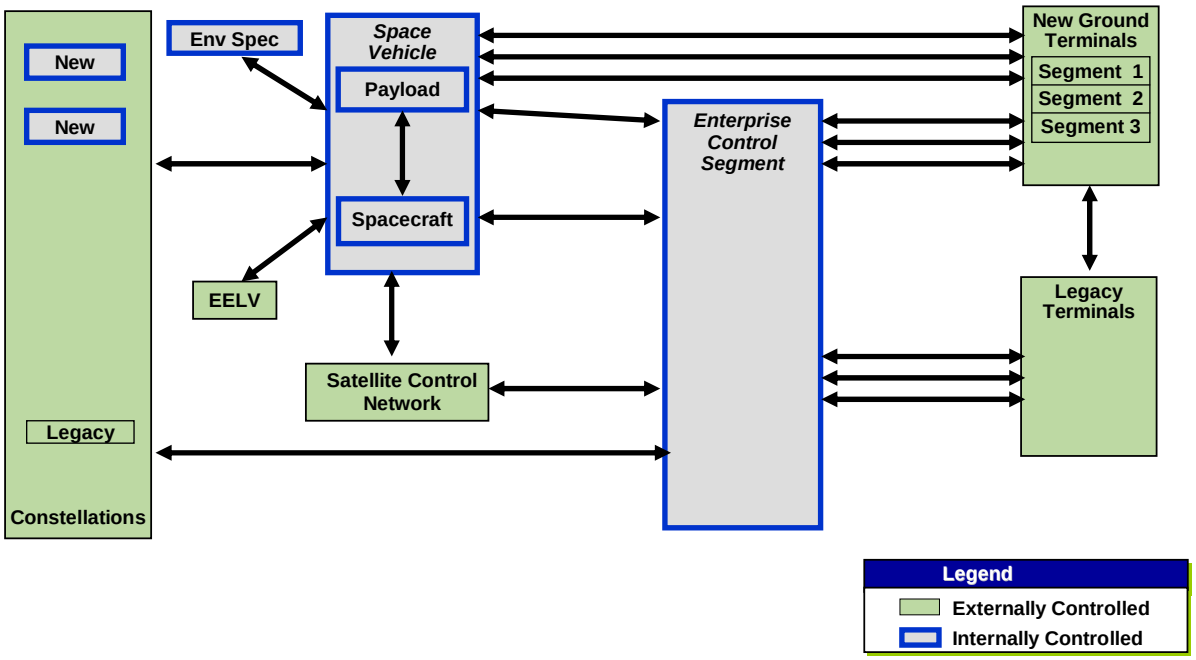

Figure 2: An Enterprise that Operates a Constellation of Space-Assets Has Many Internal and External Interfaces to Document and Control

Thus our message in this second "secret" tip is that failure to (1) write down our terms of reference and disseminate the document to all stakeholders; (2) organize and conduct a requirements working group that includes all the stakeholders; (3) manage changing requirements with discipline; and (4) diligently practice the strong principle of communication will mean that when you talk you really won't walk on the path to a successful enterprise architecture.

2.3 Not Knowing Who's Your Daddy Rabbit Is Often Another Very Bad Habit

In large enterprises of the government and commercial kinds, it often seems that everything is organized into compartments. These compartments are sometimes called "stovepipes" because they seem to have walls surrounding them and people within that focus on parochial interests without becoming cognizant of enterprise goals and priorities. There are many examples (e.g., [Wade, 2006]) in which various animal species (bats, meerkats, early-humans, etc.) protect the young of another member of their packs, clans, or tribes (organizations), or share access to a food hoard. We therefore suppose it is within human nature to protect one's own self-interests by acting consistently in terms of a stove-piped organization's roles and processes. There are at least two means of overcoming our human nature in the interest of building intelligent enterprise architectures.

Identify Process/Project Owner

Assuming that the information technology community will continue to increase the already wide-spread use of internet-exposed business systems and functionality, this tip becomes increasingly important. An interesting characteristic of web-services and *service oriented architecture* is the required orchestration of enterprise-wide and cross-enterprise web-services. [Umar, 2004] describes the environment this way: "Over the years Web technologies have merged with other distributed computing technologies and assumed a vital role in satisfying enterprise business needs." Early identification, planning and test will be vital in overcoming preexisting agreements about the use and billing of web-services. As can be seen in Figure 3, more than just the technical aspects of the XML and SOAP message passing must be considered: business processes, policy descriptions, and quality of service must all be considered and integrated to achieve an intelligent enterprise.

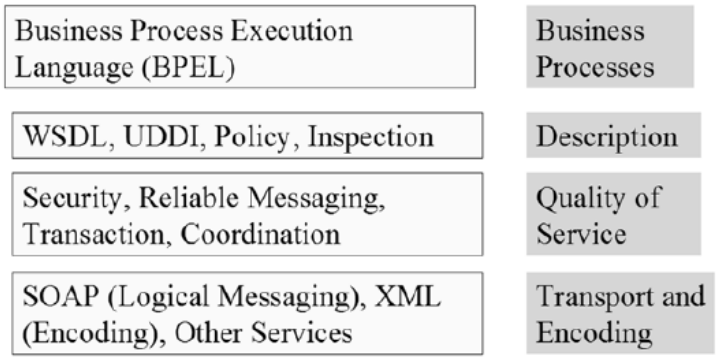

Figure 3: Evolving Web Services Complicate Architecture Integration [Umar, 2004]

A web-site (see http://www.themanager.org/Strategy/Change_Problems.htm) asserts that top management failure to support a needed change represents a barrier to efficient and effective change. Top management support is one of the critical success factors for any change effort. If top management ─ the daddy rabbit ─does not buy in, why should anybody else? It will prove difficult to instantiate a process change in an enterprise unless top management has the Merlin factor [Smith, 1994]. But that is not enough. The enterprise system architect must know who this top level manager is, and must feel confident that he can rely on that top manager to help work through the difficulties that will arise when stovepipes remain in the enterprise. This can be problematic, especially if the enterprise is the Department of Defense. Service rivalries are well documented. Yet we live in an age when the emphasis is on operations that involve more than one service and possibly involve coalition members from other nations. And commercial firms are increasingly of the international kind. Therefore, the top level manager of a joint operations enterprise should be identified at a sufficiently high level of the enterprise, and should represent herself or himself as the go-to person for any problems that may arise from service rivalries or stove-piped elements within the enterprise.

Leadership from an Enterprise Executive

Certain styles of leadership, in contradistinction to management, have been documented to prove effective when an enterprise must be transformed in the interests of its own survival. Successful leaders have been said to possess the Merlin factor [Smith, 1994]. One of us ([Carl, 2006]) has documented in the systems engineering literature the capacity of Merlin-factor leadership to successfully transform enterprises.

Secret tip number three tells us the enterprise systems architect must be able to talk problems over with the top executive that is leading the enterprise. Failure to know who this top leader is that can resolve the thorniest problems is indeed a very bad habit.

2.4 Drawing the Wrong Picture Can Become a Real Stricture

One of the principles taught to anyone who hopes to communicate with a group of people is "know your audience." (We discussed this advice in Section 2.1, above.) Another principle of this kind is "know your purpose." The two principles are related to each other in the interest of effective communication. Clearly, if you don't know the groups with whom you will communicate you are at risk of choosing the wrong level of abstraction, or the wrong words, or the wrong graphics. The wrong choices will lead to problems, not only of the political-correctness kind, but also of the sub-group complication kind. Too much detail for one group is not enough detail for another group. And the last thing a systems engineer needs to do is alienate the people to whom she or he is talking.

Drawing the correct picture was presumably part of the motivation that led John Zachman to remind us [Zachman, 1987] that there may be only one architecture, but there are multiple substantially different views of it. But complexity can be confusing even within each view so that one gets lost in the details, and too little detail can mislead. Getting the level of abstraction right is important in all the views of an architecture.

Know Your Purpose

A six step process (Figure 4) is provided in Volume III of the DoD Architecture Framework (DoDAF; [DoDAF, 2004]). All too often, inexperienced architects race to Step 5, "Build the requisite products." Although the race may satisfy a checklist to enter the next acquisition milestone decision, a MITRE technical report on lessons-learned about enterprise architecture development says, "Architectures must have a specific purpose, and should not be

developed until there is at least one agreed purpose." The report asserts that there are no generic systems or enterprises, so each effort will be highly specialized.

Related to purpose is architectural scope, Step 2. All too often we see architectures with excess detail, losing sight that various Enterprise architectural artifacts need to satisfy some useful purpose. Instead of clearly communicating, they document some quantity of a subject matter expert's knowledge.

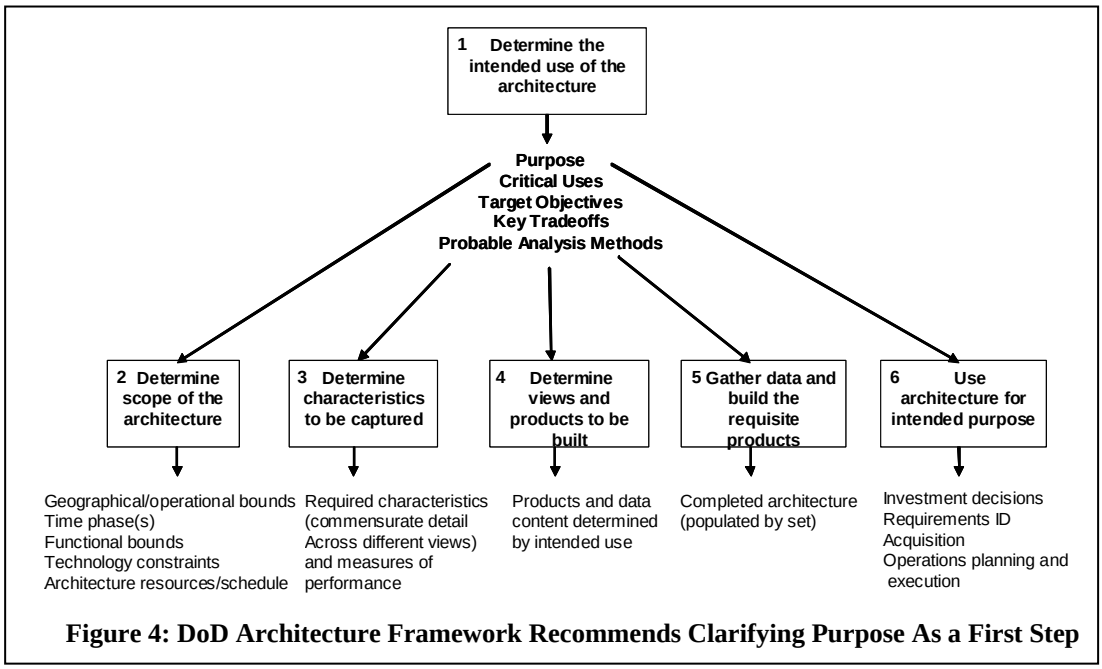

Figure 4: DoD Architecture Framework Recommends Clarifying Purpose As a First Step

All Models Are Abstractions

The point of this "secret" tip is that when a level of abstraction is too high, the model can lie; and when a level of abstraction is too low, one can get lost in the flow. It is important for the enterprise systems engineer to find the right level of abstraction that enables effective communication with the target group and is at the same time useful for its intended purpose. If one forgets the preceding remedy, "Know your Purpose", one often gets lost in great depth of detail potentially without benefit.

All models are imperfect ─ they are not the system being modeled. They represent some information about the intended system adequately for some purpose, but they omit other information that does not bear on the purpose. So there is flexibility available when constructing models; use the flexibility wisely to serve the purpose. Know your audience; know your purpose; aim your talk or presentation to the right target. Don't draw the wrong picture.

2.5 *When Only Organizations Interest You Some Will See You As Pepe Le Pew*

Within many large enterprises almost everything revolves around the organization. This focus especially dominates the DoD viewpoint, perhaps due to it's institutionally strong hierarchical structure defining the chain of command. Where are people assigned? What organizations have money, power, or influence? What organizations support other organizations? Enterprise architects should avoid what we term organizational bias. As an enterprise architect looks across a complex environment, there is much that will promote bias in the architect – operating instructions, company publications, regulations, and local procedures often tend to focus on job titles and associated job roles and responsibilities.

Because it is common to have several job titles responsible for a single activity, one must pull processes out of individual job descriptions. There can also be required activities that are not explicitly allocated to any single job title. Architects must decompose the organizations and glean the significant activities that are being accomplished. It is not an easy task for the architect to discover and map all the required processes within an organization's initial frame of reference. For this reason, a subject matter expert needs to be included in the architecting effort.

Also, inefficiencies within organizations can be easily seen once business processes are initially modeled. In one project, when examining the baseline activity tree, the team quickly saw organizational stovepipe-induced

inefficiencies. They were revealed in activities that were often repeated. In the example (Figure 5) the activity "Conduct Enroute Flight" is repeated as a child of three different parent functions in the tree. The three are associated with specific existing organizations, which could result in the development of individual systems and solutions for each, even though an organization-unique solution may not be warranted. A similar situation appears in the tree for "Conduct Air Refueling," Conduct Terminal/Objective," and "Accomplish Recovery." It is apparent that this decomposition proceeded with organizational bias. If looked at correctly, a variety of more efficient organizational constructs can be considered, and some alternatives are likely to increase unit cohesion and remove excessive coupling.

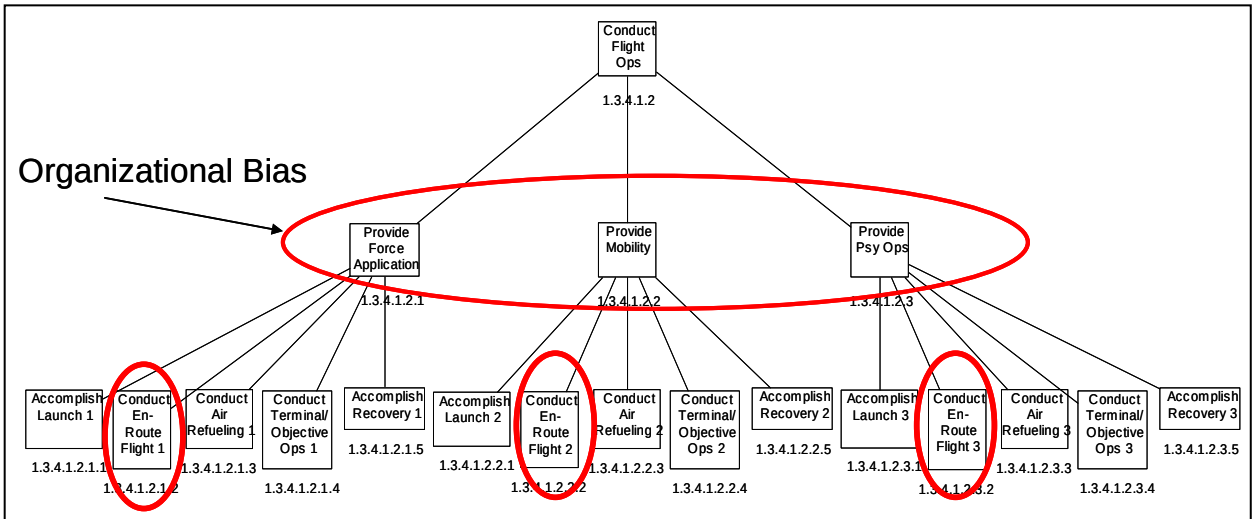

Figure 5: Organizational Bias Can Hinder Construction of an Efficient Enterprise Architecture

Leadership from a Merlin Perspective

A leader that has the Merlin perspective [Carl, 2006] can see a future, but may not know how to get to it. So that leader communicates her or his vision to a team; the team helps shape and refines the vision. The team spreads the word about the vision and its vital importance to the enterprise. The leader and her or his team empower members of the enterprise to figure out (1) what the vision means in their specific part of the enterprise, and (2) how to achieve their part of the vision. Merlin leadership is rare, but it is proven in experience to result in successful transformations of enterprises, transformations that may be critical to the continuance of the enterprise.

Enterprise Architects

Domain knowledge needs to extend beyond technical or engineering domain expertise: enterprise architects should possess operational knowledge and experience. The ability to accurately comprehend and express the important factors within a problem space as well as to accurately judge potential usefulness absolutely relies on such operational understanding and experience. Few aeronautical engineers have ever piloted the finally produced aircraft they designed. Thus, the development of a Special Operations Forces (SOF) Air Mission Planning Enterprise benefits when architected by a team that includes a SOF pilot. Thus, having four F-15E "Strike Eagle" pilots perform concept refinement and enterprise planning can successfully lead to improved time sensitive targeting in the complex DoD combat operations enterprise. Thus, having two F-15C pilots lead a team that is selecting various modifications to optimize capability and extend the life of the F-15C fleet can directly contribute to a successful project. While these examples are defense-related, operational domain knowledge is also beneficial in early commercial lifecycles, as discussed in Section 2.1, above.

2.6 Lest Planning Make a Fool of You, Keep Execution in View

Some elements of an organization are responsible for determining what, where, when and how resources will be employed to achieve strategic goals; other elements are responsible for executing the tasks that lead to tactical mission accomplishment or market success. We might call these two levels of responsibilities *planning* and *execution*. Although the planning responsibilities guide and govern the execution of tasks, the nature of the planning activities can be clearly distinguished from the execution tasks and responsibilities. Yet in today's worlds of

business and of defense, the planning and execution steps are blurring due to the growth of interconnectedness and automated processes within enterprises.

Planning Is Vacuous Without Supporting Execution

Planning primarily guides and governs the activities and the flow of resources within an enterprise, and can be strategic or tactical in nature. (Then) General Eisenhower once said "In preparation for battle I have often found plans to be useless, but planning to be indispensable." Perhaps the general felt that way because of the complexity of task execution. Planning remains vacuous until the tasks are executed by the work force that meets the enemy on the battlefield or meets competitors in the marketplace. The kinds of activities differ between planning and execution, but you can't accomplish one without the other.

An enterprise systems engineer must remove any blinders she or he is wearing by virtue of her or his organizational level of experience. You do not want planning to make a fool of you by failing to keep execution in view.

2.7 Although Movement Isn't Function, Functions Build Conjunction Junction

The models used in structured analysis to develop architectures define an oval or a rectangle to represent a function and a function is usually defined to be a process or activity that transforms its input into its output. Directed line segments represent material or information flows or movements. This is the case for both data flow diagrams and for IDEF0 diagrams. Representing data movement with a directed line segment that splits or joins may hide the need for functionality to create the splits and the joins. Figure 6 (after Figure 3.6 on page 71 of [Buede, 2000]) offers an example of how this can happen. Does input I_1 come in as a pair of lines such that one line carries $I_{1,1}$ and the other carries $I_{1,2}$, or does I_1 come in as a multiplexed stream that needs to be de-multiplexed. If the latter, functionality is implied that is not shown within this decomposition of A_0. Without access to a data dictionary or a glossary, the question remains.

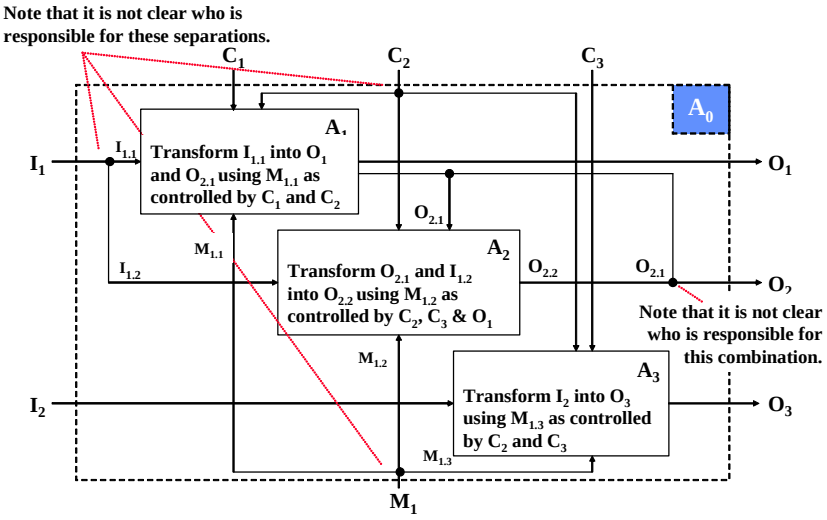

Figure 6*: Ambiguity Can Arise When Conjunction Junction Is Slighted.*

Further, it sometimes isn't clear how an enterprise systems engineer should represent data encryption and decryption and data security operations in architecture representations. Although some aspects of material, process, or information flow relate to control, which can be represented in IDEF0 diagrams, and in Enhanced Functional Flow Block Diagrams, not all such movement is control.

Define Responsibilities in a Data Dictionary or in a Glossary

Structured analysis using data flow diagrams requires definition of a separate data dictionary to define interfaces, message protocols, and so on. Structured analysis using IDEF0 diagrams requires definition of a glossary that accomplishes a similar purpose. These are two of the most commonly used system model representations. Unfortunately, the data dictionary or glossary is frequently overlooked or slighted. Yet it is worth the time and effort

to create one or the other ─ appropriate to the model being used ─ with care in order to remove the kind of ambiguity that can otherwise result.

Include "Hollow" Transfer Functions

If there was a function box within A_0 in front of the split of I_1 in Figure 7 that said "De-multiplex Data," we would have a function that transforms its input into its output in a limited sense that relates to data movement much more than to functional transformation. [Wieser, et al., 2006] calls functions that only move data *hollow* functions. A hollow function provides no meaningful transformation of the input stream. It simply takes information at location A and distributes it to location B (Figure 7).

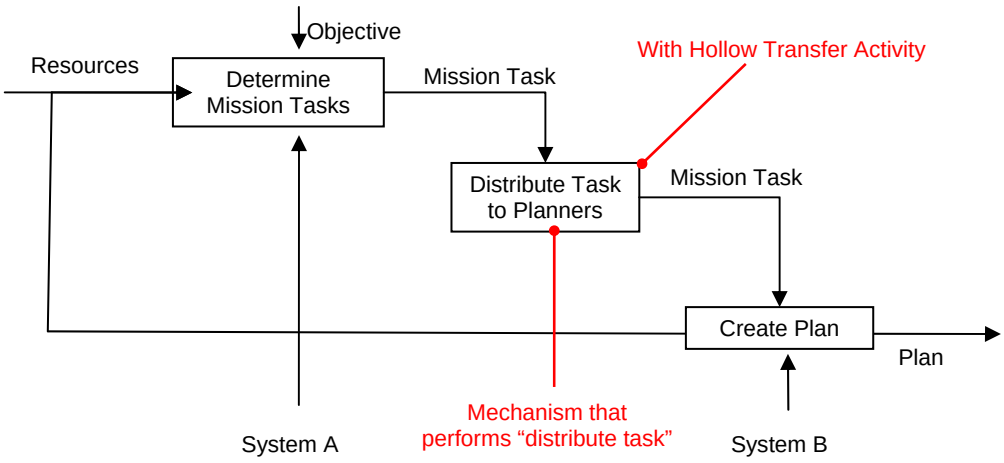

Figure 7: Hollow Transfer Activity May Usefully Be Included in Enterprise Architecture Diagrams

Although the "De-multiplex Data" function is not quite that hollow, it is similar in its functionality, and could generalize the idea of a hollow function. It makes a lot of sense to include such generalized hollow functions when trying to remove ambiguity from information-movement flows in enterprise architecture diagrams. Such generalized hollow functions can enable us to represent encryption and decryption as activities that are performed on information flows in a similar manner. And security operations are really a procedural set of activities that control aspects of an enterprise architecture, and we really do know how to represent a security operations plan as a control on functional boxes in IDEF0 diagrams; if such activities are not controls, they may be usefully represented as generalized hollow functions using data flow or IDEF0 diagrams.

The phrase that titles this "secret" tip comes from an old children's television show, *Schoolhouse Rock*. One segment of that show had cartoon characters singing, "Conjunction junction, what's your function?" And the remaining cartoon character would respond, "Hookin' up words and phrases and clauses." Hookin' up functions with data flows is in the same spirit, and just as important as understanding grammar. Making things unambiguous with data dictionaries, glossaries, and generalized hollow functions will contribute to the success of any enterprise architecture.

3.0 SUMMARY AND CONCLUSION

Adopting the seven "secret" tips presented in this paper will help assure that your enterprise architecture won't contribute to Cobb's paradox.: (1) ignorance of domain causes architecture pain, (2) if you can't talk the talk, then you can't walk the walk, (3) not knowing who's your *daddy rabbit* is often another very bad habit, (4) drawing the wrong picture can become a real stricture, (5) when only organizations interest you some will see you as Pepe Le Pew, (6) lest planning make a fool of you, keep execution in view, and (7) although movement isn't function, functions build conjunction junction. While these tips were distilled as heuristics over the course of several projects for the Department of Defense, their remedies derive in part from commercial experience. The tips should therefore appeal widely to systems engineers and enterprise architects from both worlds. After all, intelligence in, intelligence out.

4.0 REFERENCES

[Belsey, 2002] Catherine Belsey, *Poststructuralism: A Very Short Introduction*: Oxford, UK, Oxford University Press, 2002, pages 7-8.

[Buede, 2000] Dennis M. Buede, *The Engineering Design of Systems: Models and Methods*, NY: John Wiley & Sons, 2000.

[Carl, 2005] Joseph W. Carl, "Why Are Requirements So Hard To Get Right?" *Proc. 15th Annual INCOSE International Symp.*: Toulouse, France, June 2005.

[Carl, 2006] Joseph W. Carl, "A Merlin Perspective Shines Light on Tough Problems," *Proc. 16th Annual INCOSE International Symp.*: Orlando, Florida, 2006.

[Cobb, 2004] Martin Cobb, Treasury Board of Canada Secretariat, spoken at CHAOS University, a Standish Group, education organization , 2004; see http://www.standishgroup.com/sample_research/unfinished_voyages_1.php.

[Colombi, 2006] J. Colombi, L. Anderson, P Doty, M. Griego, K. Timko, B Hermann, "Operational Domain Systems Engineering," Insight, vol. 9, no. 1 (October 2006), pgs. 23-24.

[DoDAF, 2004] DoD Architecture Framework Version 1.0, Volume III: Deskbook, 9 February 2004 http://www.defenselink.mil/nii/doc/DoDAF_v1_Volume_II.pdf

[DoD-CIO, 2005] The Department of Defense Chief Information Officer, "Communities of Interest in the Net-Centric DoD," Version 1.0, May, 2004.

[Grady, 2006] Jeffrey O. Grady, "The Ultimate Foundation of Systems Engineering," InSight, Vol 8, No (March 2006).

[Insight, 2006] Insight Events, "The 2007 International Symposium is coming!" *Insight, vol. 9, no. 1* (October 2006), p.38.

[Kasser, 2002] Joseph Kasser, "A prototype tool for improving the wording of requirements," *Proc. 12th Annual INCOSE International Symposium*: July 2002.

[Smith, 1994] Charles E. Smith, "The Merlin Factor: Leadership and Strategic Intent," *Business Strategy Review, Vol. 5, Issue 1*, pp. 67-83, Oxford, UK: Oxford University Press, Spring 1994.

[Snow, 1998] C. P. Snow, *The Two Cultures*: Cambridge, England, Cambridge University Press: 1998 (originally published as the Rede Lecture: Cambridge, England, Cambridge University Press: 1959).

[Umar, 2004], Amjad Umar, "The Emerging Role of the Web for Enterprise Applications and ASPs", *Proceedings of the IEEE*, vol. 92, no. 9, September, 2004.

[Wade, 2006] Nicholas Wade, *Before the Dawn: Recovering the Lost History of Our Ancestors*, New York, NY, Penguin Press, 2006.

[Wieser, 2006] T. Wieser, J. Miller, A. Piepkorn, J. Kennedy, R. Mills, and J. Colombi, "Heuristics for Joint Architecting," *Defense AT&L*, Nov.-Dec., 2006, pp. 2-6.

[Zachman, 1987] John Zachman, "A framework for information systems architecture," IBM Systems Journal, vol. 26, no. 3, pp. 276-292, 1987.

5.0 BIOGRAPHIES

Joseph W. Carl

Joe began his systems engineering career on active duty with the US Air Force as the Chief Engineer on the Avionics Upgrade Program for the F/FB-111 fleet. He next worked as a systems engineer for Harris Corporation. He took a job with Riverside Research Institute in September, 2006, and was assigned to teach at the Air Force Center for Systems Engineering. Joe has a PhD from the Ohio State University, is a PE registered in Ohio, and is an INCOSE certified systems engineering professional. He is a senior member of the IEEE, a member of PMI, and an active member of INCOSE, where he serves as a co-chair of the Anti-Terrorism International Working Group, as a member of the Object-Oriented Systems Engineering Methods Working Group, as a member of the Information Systems Working Group, and has served as a task leader on the SE Handbook v.3 Working Group, and as a past president of the Chesapeake Chapter. Joe completed the Air Force Marathon in 2006 along with 8,000 others.

John M. Colombi

Lt Col John Colombi is an Assistant Professor of Electrical Engineering working in the Air Force Center for Systems Engineering at the Air Force Institute of Technology. He teaches and leads sponsored research in support

of the Systems Engineering program. Before joining the faculty, Dr. Colombi led C4ISR systems engineering, architecture and integration activities including Chief of Systems Engineering for US Airborne Warning and Control System (AWACS) at Hanscom AFB. Before that, he had served at the National Security Agency developing information system security. His first Air Force assignment at the Air Force Research Laboratory ran communications networking and network management research. He is a member of both INCOSE and IEEE. He does not run marathons.